arXiv:1303.6867v1 [cs.DS] 27 Mar 2013

Parameterized algorithms for the 2-clustering problem with minimum sum and minimum sum of squares objective functions (Extended abstract)

Bang Ye Wu*and Li-Hsuan Chen
Dept. of Computer Science and Information Engineering
National Chung Cheng University, Taiwan

Abstract

In Min-Sum 2-Clustering *problem, also known as 2-cluster graph editing and correlation 2-clustering, we are given a graph $G = (V, E)$ and a parameter k; and the goal is to determine if there exists a 2-partition of V such that the total conflict number is at most k, in which the conflict of a vertex is the number of its non-neighbors in the same cluster and neighbors in the different cluster. In this paper we show a parameterized algorithm with time complexity $O(n \cdot 2.618^{r/(1-4r/n)} + n^3)$, in which $r = k/n$. Particularly, the time complexity is $O^*(2.618^{k/n})$ for $k \in o(n^2)$ and polynomial for $k \in O(n \log n)$. We also design a parameterized algorithm for a variant which minimizes the sum of squared conflicts. For $k \in o(n^3)$, the algorithm runs in $O(n^3 \cdot 5.171^{\theta})$ time, in which $\theta = \sqrt{k/n}$.*

Key words. parameterized algorithm, kernelization, cluster graph, clustering, graph modification.

AMS subject classifications. 65W05, 68R10, 68Q25, 05C85, 91C20

1 Introduction

Problem definition and motivation. Clustering is an important problem with applications in numerous fields, and the *cluster graph editing* problem, also known as *correlation clustering* is a graph theoretic approach to clustering [2, 22]. A *cluster graph* is a graph composed of disjoint maximal cliques. The cluster graph editing problem asks for a minimum number of edge insertions and deletions to modify the input graph into a cluster graph. As a variant, the *p-cluster graph editing* problem is to modify the input into exactly p disjoint cliques [22]. The cluster graph editing problem focuses on the editing set, i.e., the edges to be inserted and deleted. As

*National Chung Cheng University, ChiaYi, Taiwan 621, R.O.C., E-mail: bangye@cs.ccu.edu.tw

in the correlation clustering problem, an editing edge can also be thought of as a *disagreement* in the clustering. If we change the point of view from edges to vertices, an editing edge represents a deficiency on its endpoints in the clustering. Let *conflict number* of a vertex be the number of editing edges incident to it. Then, an editing set of cardinality k one-to-one corresponds to a clustering with total conflict number $2k$. That is, the cluster graph editing problem is equivalent to finding a vertex partition with minimum total conflict. The transformation of problem definition provides us easier ways to define other meaningful objective functions on the conflict numbers, such as sum of squares.

In this paper we focus on 2-clusterings. For a set V, a *2-partition* of V is an unordered pair $\pi = (V_1, V_2)$ such that $V_1 \cap V_2 = \emptyset$ and $V_1 \cup V_2 = V$. In this paper we assume $V_1, V_2 \neq \emptyset$. We say that V_1 and V_2 are the two clusters corresponding to π. For a 2-partition π of V, two vertices u and v *conflict* with each other if they are in the same cluster but $(u, v) \notin E$ or they are in different clusters but $(u, v) \in E$. Let $C_\pi(v)$ denote the set of vertices conflicting with v in π and $c_\pi(v) = |C_\pi(v)|$ be the *conflict number* of v. Clearly $u \in C_\pi(v)$ if and only if $v \in C_\pi(u)$. For an input graph, a 2-clustering problem in general asks for a 2-partition such that the conflicts are as small as possible. In this paper we study the 2-clustering problem with two natural objective functions: minimizing the sum of conflicts and the sum of squared conflicts. Precisely speaking, let $h_1(\pi) = \sum_{v \in V} c_\pi(v)$ and $h_2(\pi) = \sum_{v \in V} c_\pi^2(v)$. The problems are formally defined as follows. We shall focus on their decision versions.

> Problem: Min-Sum 2-Clustering
> Instance: A graph $G = (V, E)$ and a nonnegative integers k.
> Question: Is there a 2-partition π of V such that $h_1(\pi) \leq k$.

The second problem, named Min-Square 2-Clustering, is defined similarly, except that the objective function is h_2.

Previous results. Shamir et al. [22] studied the computational complexities of three edge modification problems. Cluster Editing asks for the minimum total number of edge insertions and deletions to modify a graph into a cluster graph, while in Cluster Deletion (respectively, Cluster Completion), only edge deletions (respectively, insertions) are allowed. They showed that Cluster Editing is NP-hard, Cluster Deletion is Max SNP-hard, and Cluster Completion is polynomial-time solvable. They also showed that p-Cluster Deletion is NP-hard for any $p > 2$ but polynomial-time solvable for $p = 2$, and p-Cluster Editing is NP-hard for any $p \geq 2$.

An instance of a parameterized problem consists of (I, k), where k is the parameter. A problem is *fixed-parameter tractable* (FPT) if the problem can be solved in time complexity $O(f(k) \cdot p(|I|))$, where f is an arbitrary computable function of k and p is a polynomial in the input size. For more details about parameterized complexity, we refer to the book of Downey and Fellows [11]. *Kernelization* is a widely-used technique for parameterized algorithms. Usually, by some designed *reduction rules*, a kernelization algorithm converts an instance (I, k) to a reduced instance (I', k'), called a *kernel* such that the answer is not changed, $k' \leq k$ and $|I'|$ is bounded by a computable function of k.

The parameterized version of Cluster Editing, and variants of it, were studied intensively [3, 4, 7, 9, 10, 16, 17, 18]. A variant with vertex (rather than edge) deletions was considered in [20], and another variant in which overlapping clusters are allowed was studied in [12].

The correlation clustering problem (on complete signed graphs) was formulated and studied in [2], in which the authors presented a PTAS for the maximization version and a constant factor approximation algorithm for the minimization version. Ailon et al. [1] showed some approximation results for the minimization version, including the unweighted case and the weighted case with some interesting assumptions on weights. Giotis and Guruswami showed that both the minimization and the maximization versions of the p-Correlation Clustering problem admit PTAS for the unweighted case [15]. The maximization 2-Correlation Clustering problem is also known as Balanced Subgraph which name comes from the application in social network analysis [19, 23].

For p-Cluster Editing, a kernel with $(p+2)k+p$ vertices was given by Guo [18]. A variant such that the conflict number of each vertex must be bounded by a parameter was studied in [21]. The problem of finding a 2-clustering minimizing the maximum conflict number has been shown NP-hard [8]. Very recently, Fomin et al. gave a parameterized algorithm with time complexity $O(2^{O(\sqrt{pk})} + n^2)$, where n is the number of vertices [14]. They also showed a lower bound for the parameterized complexity: there exists $p = \Theta(k^\sigma)$ for any constant $0 \leq \sigma \leq 1$ such that it is very unlikely to solve the problem with time complexity $2^{o(\sqrt{pk})} \cdot n^{O(1)}$.

Our contributions. Recall that Min-Sum 2-Clustering with parameter $2k$ is equivalent to 2-Cluster Editing with parameter k. We should first note that it is not hard to solve Min-Sum 2-Clustering in polynomial time for $k \in O(n)$. As an extreme example, if $k < n$, by pigeonhole principle, there must be a vertex whose neighborhood in the solution is the same as in the input graph, and thus the solution can be found by trying $(N_G[v], V(G) \setminus N_G[v])$ for all $v \in V(G)$, where G is the input graph and $N_G[v]$ denotes the closed neighborhood. This observation can be easily extended to $k \in O(n)$ and leads to a polynomial-time algorithm. Therefore, for Min-Sum 2-Clustering, a better result should be expected.

In this paper we develop parameterized algorithms for Min-Sum 2-Clustering and Min-Square 2-Clustering. We design a kernelization algorithm for Min-Sum 2-Clustering. First, the problem is equivalent to finding a flipping set of a bounded size for an initial vertex 2-partition, i.e., determining which vertices should be swapped to the other cluster. By a precise analysis, it is shown that the kernel size is at most $2k/(n-4f)-f$ if there is a flipping set of size f. The kernelization algorithm iteratively decreases the flipping quota f and flips the vertices with too many conflicts until the kernel size fits the bound. Then, a search-tree algorithm is employed to determine if there exists any flipping set of size bounded by the remaining flipping quota. By analyzing the worst case of the remaining flipping quota, we show that the problem can be solved in $O(n \cdot 2.618^{r/(1-4r/n)} + n^3)$ time, in which $r = k/n$. Particularly, the time complexity is $O^*(2.618^{k/n})$ for $k \in o(n^2)$ and polynomial for $k \in O(n \log n)$, where $O^*(\cdot)$ is as $O(\cdot)$ ignoring polynomial factors. We also note that this result is better than $O^*(2^{O(\sqrt{pk})})$ recently

obtained by Fomin et al. [14] for the special case of $p = 2$. Even when $k = \delta n^2$ with small constant δ, our algorithm improves the brute-force algorithm significantly. For example, when $\delta = 0.1$, the time complexity is $O^*(2^{0.231n})$, much better than $O^*(2^n)$.

The other contribution is an $O(n^3 \cdot 5.171^\theta)$-time algorithm for Min-Square 2-Clustering, where $\theta = \sqrt{k/n}$ and $k \in o(n^3)$. The result is obtained by a similar method and some observations on the optimal solutions.

Organization of the paper. In Section 2 we give some notation and definitions, as well as some properties, used in this paper. The reduction algorithm is in Section 3. In Sections 4 and 5 we show the algorithms for the two problems, respectively. Finally some concluding remarks are in Section 6.

2 Preliminaries

For two sets S_1 and S_2, the set difference is denoted by $S_1 \setminus S_2$, and the symmetric difference is denoted by $S_1 \ominus S_2 = (S_1 \setminus S_2) \cup (S_2 \setminus S_1)$. For simplicity, $S_1 \ominus v = S_1 \ominus \{v\}$ for $v \in S_1$. For a graph G, the vertex set and the edge set are denoted by $V(G)$ and $E(G)$, respectively. For $S \subset V(G)$, the subgraph induced by S is denoted by $G[S]$. Throughout this paper, $G = (V, E)$ is the input graph and $n = |V|$. The terms "2-partition" and "conflict" have been defined in the previous section. For $S \subset V$, let $C_\pi(v, S) = C_\pi(v) \cap S$ and $c_\pi(v, S) = |C_\pi(v, S)|$. For disjoint S_1 and S_2, $c_\pi(S_1, S_2) = \sum_{v \in S_1} c_\pi(v, S_2)$. Note that $c_\pi(S_1, S_2) = c_\pi(S_2, S_1)$. When there is no confusion, we shall omit the subscript and simply use $c(\cdot)$ instead of $c_\pi(\cdot)$.

In the literature, Min-Sum 2-Clustering is also known in another formulation: 2-Cluster Editing. A graph $G = (V, E)$ is a *2-cluster graph* if it consists of exactly two disjoint cliques. That is, there exists a 2-partition $\pi = (V_1, V_2)$ of V such that both $G[V_1]$ and $G[V_2]$ are maximal cliques. For $G = (V, E)$, a set $D \subset V \times V$ is an *editing set* (to 2-cluster graph) for G if $G' = (V, E \ominus D)$ is a 2-cluster graph. In other words, G can be modified into a 2-cluster graph by inserting $D \setminus E$ and deleting $D \cap E$. Given a graph G and integer k, the 2-cluster graph editing problem asks if there is an editing set D for G such that $|D| \leq k$. Let $E^*(\pi) = \bigcup_{i=1,2} \{(u, v) | u, v \in V_i\}$ which is the edges set of the 2-cluster graph. By definition, if D is an editing set and π is the corresponding 2-partition, we have $E \ominus D = E^*(\pi)$ and $D = \bigcup_{v \in V} \{(v, u) | u \in C_\pi(v)\}$. Therefore, finding an editing set is equivalent to finding the corresponding 2-partition. Furthermore, $|D| = (1/2) \sum_{v \in V} c_\pi(v)$. Consequently the problem is equivalent to Min-Sum 2-Clustering.

To *flip* a vertex v in a 2-partition π is to move v to the other cluster, that is, we change π to $\pi \ominus v \equiv (V_1 \ominus v, V_2 \ominus v)$. Flipping a vertex subset S changes (V_1, V_2) to $(V_1 \ominus S, V_2 \ominus S)$. If $\pi' = \pi \ominus v$, then $C_{\pi'}(v) = V \setminus \{v\} \setminus C_\pi(v)$. That is, flipping a vertex exchanges its conflicts and non-conflicts. Furthermore, for a flipping set F, only those conflicts in $F \times (V \setminus F)$ change. If $\pi' = \pi \ominus F$, then

$$C_{\pi'}(v) = \begin{cases} C_\pi(v, F) \cup (\bar{F} \setminus C_\pi(v, \bar{F})) & \text{if } v \in F \\ C_\pi(v, \bar{F}) \cup (F \setminus C_\pi(v, F)) & \text{if } v \in \bar{F} \end{cases} \tag{1}$$

in which $\bar{F} = V \setminus F$.

It is convenient to represent the conflicts as a *conflict graph*. The conflict graph of π is a simple undirected graph H with $V(H) = V$ and $E(H) = \bigcup_{v \in V} \{(v, u) | u \in C_\pi(v)\}$. If H' is the conflict graph of $\pi' = \pi \ominus F$, we have

$$E(H') = E(H[F]) \cup E(H[\bar{F}]) \cup \{(u, v) | u \in F, v \in \bar{F}, (u, v) \notin E(H)\}. \quad (2)$$

The *profit* of a flipping set F, denoted by $\Delta(F)$, is the decrement of total conflict after flipping F, i.e., $\Delta(F) = h_1(\pi) - h_1(\pi \ominus F) = \sum_v c_\pi(v) - \sum_v c_{\pi \ominus F}(v)$.

Lemma 1: $\Delta(F) = 4c_\pi(F, \bar{F}) - 2|F||\bar{F}|$.

Corollary 2 : $\Delta(\{v\}) = 4c_\pi(v) - 2(n - 1)$.

3 Reduction algorithm

In this section, we propose a preprocessing algorithm which reduces the instance size. Consider the following problem.

> INSTANCE: A graph $G = (V, E)$, a vertex $s \in V$, and nonnegative integers K, f, and t.
> QUESTION: Is there a 2-partition π of V such that $\sum_v c_\pi(v) \leq K$, $c_\pi(s) \leq f$ and $c_\pi(v) \leq t$ for any other vertex $v \in V$?

Definition 1: Let π be a 2-partition of V and K, t be nonnegative integers. A vertex subset F is a (K, t, f)-*feasible* flipping set for π if $|F| \leq f$, $\sum_v c_{\pi'}(v) \leq K$, and $c_{\pi'}(v) \leq t$ for any other vertex $v \in V$, where $\pi' = \pi \ominus F$.

We assume $t + 2f \leq \varepsilon n$ for some constant ε. Let $\pi_s = (N_G[s], V \setminus N_G[s])$ in which $N_G[s]$ is the closed neighborhood of s in G. The above problem is equivalent to determining a (K, t, f)-feasible flipping set for π_s.

The reduction algorithm outputs a 2-partition π, a vertex subset $U \subset V$ and an integer $m \leq f$ such that there exists a (K, t, f)-feasible flipping set in V for π_s if and only if there exists a (K, t, m)-feasible flipping set in U for π. We shall call f the *flipping quota*. To avoid confusion, we use different symbols f and m for the flipping quotas before and after the reduction algorithm, respectively. The algorithm is based on the following reduction rules for any 2-partition π and integer f.

R1: For any vertex v, if $c_\pi(v) > t + f$, v must be flipped.

R2: For any vertex v, if $c_\pi(v) < n - t - f$, v cannot be flipped.

R3: If $f = 0$ and there exists v with $c_\pi(v) > t + f$, there is no any feasible flipping set.

R4: If $c_\pi(v) \leq t + f$ for all $v \in V$ and $|U| > K/(n - t - 2f) - f$, there is no any feasible flipping set of size f, in which $U = \{v | c_\pi(v) \geq n - t - f\}$.

In the following we shall show the correctness of the reduction rules. We start from the first three simple rules.

Lemma 3: The reduction rules R1–R3 are correct.

By R2, we can put those vertices with $c_\pi(v) < n-t-f$ into a set X, and when none of the first three rules can be applied, $U = V \setminus X$ is the set of *undetermined vertices*. We note that, once a vertex is flipped, it should be put into X. But in fact we do not even need to divide V into X and U until R1 is no more applicable. The rule R4 is an upper bound of $|U|$. We now show its correctness. Suppose that $F \subset U$ is a feasible flipping set. Let $\bar{F} = V \setminus F$. In the remaining paragraphs of this section we omit the subscript π.

We start from a simple but weaker bound. Since $c(v) \leq t+f$, by Lemma 1, the profit of F is $\Delta(F) = 4c(F,\bar{F}) - 2|F||\bar{F}| \leq 4f(t+f) - 2f(n-f) = (4t-2n)f + 6f^2$. Immediately, if $\sum_v c(v) > K + (4t-2n)f + 6f^2$, there does not exist any feasible flipping set. Since $c(v) \geq n-t-f$ for any $v \in U$,

$$(n-t-f)|U| \leq \sum_{v \in U} c(v) \leq \sum_{v \in V} c(v) \leq K + (4t-2n)f + 6f^2,$$

and therefore

$$|U| \leq \frac{K + (4t-2n)f + 6f^2}{n-t-f}. \tag{3}$$

We shall show the better bound in R4 by a more precise analysis.

Lemma 4: If there exists a feasible flipping set of size f, then $|U| \leq K/(n-t-2f) - f$.

We note that the upper bound is for the case of flipping set of size exactly f. As f decreases, the upper bound increases but $|U|$ decreases because possibly more vertices are flipped. Since our goal is to find a flipping set of size at most f, the algorithm iteratively decreases f until the bound is satisfied. The reduction algorithm is in Algorithm 1.

Lemma 5: The algorithm REDUCTION takes $O(n^2)$ time.

4 Minimum sum of conflicts

In this section we show a parameterized algorithm for MIN-SUM 2-CLUSTERING. We start from two simple properties.

Fact 6: If π is a 2-partition with minimum $h_1(\pi)$, then $c_\pi(v) \leq (n-1)/2$ for any v.

The next property comes from the definitions.

Fact 7: For any 2-partition π, there exists a vertex v with conflict number at most $h_1(\pi)/n$.

Algorithm 1 : REDUCTION(G, s, K, t, f)

Input: a graph $G = (V, E)$, a vertex $s \in V$, and integers K, t and f.
Output: a 2-partition π, a vertex subset U, and an integer m.

initially $\pi = (N_G[s], V \setminus N_G[s])$;
while $\exists v$ such that $c_\pi(v) > t + f$ **do**
    $\pi \leftarrow \pi \ominus v$; //flipping v
    update $c_\pi(u)$ for each u;
    $f \leftarrow f - 1$;
    if $f < 0$ **then**
        report "No" and terminate; // R3, no solution
    end if
end while
let $U = \{v | c_\pi(v) \geq n - t - f\}$;
if $|U| > \frac{K}{n-t-2f} - f$ **then**
    $f \leftarrow f - 1$;
    if $f < 0$ **then** report "No" and terminate;
    goto Step 2;
end if
return $(\pi, U, m = f)$;

By the above two facts, we can focus on determining a 2-partition π such that $h_1(\pi) = \sum_v c_\pi(v) \leq k$, $c_\pi(s) \leq k/n$ for some $s \in V$, and $c_\pi(v) \leq (n-1)/2$ for each $v \in V$. Hence we can use the reduction algorithm with $K = k$, $t = (n-1)/2$ and $f = k/n$. The remaining work is to check if $h_1(\pi \ominus F) \leq k$ for every possible flipping set F of size at most m, in which m is the flipping quota returned by the reduction algorithm. This work can be done by a simple search-tree algorithm on the reduced instance. The search-tree algorithm picks up an arbitrary undetermined vertex v and recursively solves the problem for two cases: flipping v or not. To make the reduced instance a kernel, the time complexity of the search-tree algorithm should depend on the kernel size rather than the input size. Fortunately, for MIN-SUM 2-CLUSTERING, it can be easily done by the following method.

Let U be the set of undetermined vertices and $X = V \setminus U$. For each $u \in U$, we record $c_\pi(u, X)$. Also we record $\chi = c_\pi(X, X)$ and $|X|$. When moving a vertex u from U to X, we update χ to $\chi + 2c_\pi(u, X)$ and also update $c_\pi(v, X)$ to $c_\pi(v, X) + c_\pi(u, v)$ for each $v \in U \setminus \{u\}$. When flipping and removing u, we update χ to $\chi + 2(|X| - c_\pi(u, X))$ and also update $c_\pi(v, X)$ to $c_\pi(v, X) + 1 - c_\pi(u, v)$ for each $v \in U \setminus \{u\}$. This work can be done in $O(|U|)$ time if the conflict graph induced by U is used as the input of the search-tree algorithm. Since the number of conflict edges is $O(|U|^2)$, each recursive call takes $O(|U|^2)$ time.

The whole algorithm is in Algorithm 2. Note that we need to try every vertex v as the input of REDUCTION.

In the remaining paragraph of this section we shall show the time complexity of Algorithm 2. The most important thing is the time complexity of the search-tree algorithm.

Lemma 8: The search-tree algorithm runs in $O(\phi^{|U|+m})$ time, in which $\phi =$

Algorithm 2 : Finding min-sum 2-clustering

Input: a graph $G = (V, E)$ and integer k.
Output: a 2-partition π with $h_1(\pi) \leq k$ or output "No".

```
1: for each v ∈ V do
2:     call REDUCTION(G, v, k, (n − 1)/2, k/n) to compute (π, U, m);
3:     if REDUCTION reports "No" then
4:         goto step 1;
5:     end if
6:     construct the conflict graph H induced by U;
7:     call search-tree algorithm with m, k, and H as input;
8:     if a feasible solution is found then
9:         output the answer and terminate;
10:    end if
11: end for
12: report "No";
```

$\frac{1+\sqrt{5}}{2} \approx 1.618$.

Theorem 9: MIN-SUM 2-CLUSTERING can be solved in $O(n \cdot 2.618^{r/(1-4r/n)} + n^3)$ time, in which $r = k/n$.

When $k \in o(n^2)$, since $m \leq k/n$, $(4m/n)(k/n) < \varepsilon k/n$ for any constant $\varepsilon > 0$ when n is sufficiently large. Then

$$|U| \leq \frac{k}{n/2 - 2m} - m = \frac{2k/n}{1 - 4m/n} - m < (2 + \varepsilon)(k/n) - m. \tag{4}$$

Since ε can be arbitrarily small, $\phi^{(2+\varepsilon)} \approx \phi^2 \approx 2.618$.

Corollary 10 : When $k \in o(n^2)$, MIN-SUM 2-CLUSTERING can be solved in $O(n \cdot 2.618^{k/n} + n^3)$ time.

Corollary 11 : MIN-SUM 2-CLUSTERING can be solved in polynomial time if $k \in O(n \log n)$.

When $k \in \Theta(n^2)$, the time complexity can be expressed as follows, in which the condition $\delta < 0.186$ is to ensure the result is better than the naive $O^*(2^n)$-time algorithm.

Corollary 12 : When $k = \delta n^2$ with $\delta < 0.186$, MIN-SUM 2-CLUSTERING can be solved in $O(n \cdot 2.618^{\delta n/(1-4\delta)} + n^3)$ time.

5 Minimizing the sum-of-squares

Recall that $h_2(\pi) = \sum_v c_\pi^2(v)$ is the sum of squared conflicts for a 2-partition π. Given a graph G and an integer k, MIN-SQUARE 2-CLUSTERING determines if there exists a 2-partition π with $h_2(\pi) \leq k$. In this section, we show a parameterized algorithm with parameter k.

Lemma 13: If $h_2(\pi) \leq k$, then $\sum_v c_\pi(v) \leq \sqrt{nk}$ and there exists a vertex v with $c_\pi(v) \leq \sqrt{k/n}$.

Lemma 14: If π is a 2-partition with minimum $h_2(\pi)$, then $c_\pi(v) \leq \sqrt{n(n-1)/2}$ for any vertex v.

The parameterized algorithm for MIN-SQUARE 2-CLUSTERING is similar to the one in the previous section. By the above two lemmas, we try every vertex as the one with conflict quota at most $f = \sqrt{k/n}$, and for each iteration use REDUCTION with total conflict $K = \sqrt{nk}$, individual bound $t = \sqrt{n(n-1)/2}$ and flipping quota $f = \sqrt{k/n}$. For the reduced instance U and m, a search-tree algorithm is employed to check all possible flipping sets. But, unfortunately, we did not find a way to compute the objective function h_2 with time complexity only depending on $|U|$. Therefore it takes $O(n^2)$ time to compute h_2 for each flipping set.

Theorem 15: For $k \in o(n^3)$, MIN-SQUARE 2-CLUSTERING can be solved in $O(n^3 \cdot \phi^{3.414\theta}) \approx O(n^3 \cdot 5.171^\theta)$ time, in which $\theta = \sqrt{k/n}$.

Corollary 16 : For $k \in O(n \log^2 n)$, MIN-SQUARE 2-CLUSTERING can be solved in polynomial time.

Similar to MIN-SUM 2-CLUSTERING, we can also derive the time complexity for $k = \Theta(n^3)$. But it is better than the brute-force algorithm only when k/n^3 is very small.

Corollary 17 : For $k = \delta^2 n^3$, MIN-SQUARE 2-CLUSTERING can be solved in $O(n^3 \cdot \rho^n)$, in which $\rho = \phi^{\frac{(2+\sqrt{2})\delta}{1-(4+2\sqrt{2})\delta}}$.

6 Concluding remarks

In this paper, we show parameterized algorithms for 2-clusterings with minimum conflict and minimum sum of squared conflicts. The minimum conflict problem is equivalent to the 2-cluster graph editing problem in the literature. As in many other optimization problems, the sum-of-squares objective function usually provides solutions with both small total amount and small individual amounts. The proposed algorithms significantly improve the brute-force algorithm when k is relatively small, i.e., $k \in o(n^2)$ for MIN-SUM 2-CLUSTERING and $k \in o(n^3)$ for MIN-SQUARE 2-CLUSTERING.

The time complexity in Lemma 8 is shown by induction, which is then used to derive the time complexities of the two algorithms. One may wonder if the bound is over estimated. In the appendix, we show the bound is tight when k is relatively small.

A straightforward interesting question is how to generalize the algorithms to the case of more than two clusters. However, the problem seems to become much more difficult when the number of clusters is more than two. There are also some other interesting objective functions, such as min-max conflict, minimum weighted sum. Another type of interesting problems is to relax the definition of cluster, such as minimum editing to k-plex, minimum editing to cluster graph with bounded conflict.

References

[1] N. Ailon, M. Charikar, and A. Newman, Aggregating inconsistent information: Ranking and clustering, J. ACM 55(5):1–27, 2008.

[2] N. Bansal, A. Blum, and S. Chawla, Correlation clustering, Machine Learning, Special Issue on Clustering 56:89–113, 2004.

[3] S. Böcker and P. Damaschke, Even faster parameterized cluster deletion and cluster editing, Inf. Process. Lett. 111(14):717–721, 2011.

[4] S. Böcker, S. Briesemeister, Q.B.A. Bui, A. Truss, Going weighted: Parameterized algorithm for cluster editing, Theor. Comput. Sci. 410 (52):5467–5480, 2009.

[5] P. Bonizzoni, G. Della Vedova, and R. Dondi, A ptas for the minimum consensus clustering problem with a fixed number of clusters, in: Proc. Eleventh Italian Conference on Theoretical Computer Science, 2009.

[6] P. Bonizzoni, G. Della Vedova, R. Dondi, and T. Jiang, On the approximation of correlation clustering and consensus clustering, J. Comput. System Sci. 74(5): 671–696, 2008.

[7] J. Chen, J. Meng, A $2k$ kernel for the cluster editing problem, in: M.T. Thai, S. Sahni (Eds.), Computing and Combinatorics Conf., COCOON 2010, in: LNCS, vol. 6196, 2010, pp. 459–468.

[8] L.-H. Chen, M.-S. Chang, C.-C. Wang, B. Y. Wu, On the Min-Max 2-Cluster Editing Problem, in R.-S. Chang, L. C. Jain, S.-L. Peng (Eds.), Advances in Intelligent Systems and Applications, Vol. 1, in: Smart Innovation, Systems and Technologies, vol. 20, 2013, pp. 133–142.

[9] P. Damaschke, Bounded-degree techniques accelerate some parameterized graph algorithms, in: J. Chen, F.V. Fomin (Eds.), Int. Workshop on Parameterized and Exact Comp., IWPEC 2009, in: LNCS, vol. 5917, 2009, pp. 98–109.

[10] P. Damaschke, Fixed-parameter enumerability of cluster editing and related problems, Theory Computing Syst. 46:261–283, 2010.

[11] R. G. Downey and M. R. Fellows, Parameterized Complexity, Springer-Verlag, New York, 1999.

[12] M. R. Fellows, J. Guo, C. Komusiewicz, R. Niedermeier, and J. Uhlmann, Graph-based data clustering with overlaps, Discrete Optim. 8(1):2–17, 2011.

[13] V. Filkov and S. Skiena, Integrating microarray data by consensus clustering, International Journal on Artificial Intelligence Tools 13(4): 863–880, 2004.

[14] F. Fomin, S. Kratsch, M. Pilipczuk, M. Pilipczuk and Y. Villanger, Tight bounds for parameterized complexity of cluster editing, to appear in STACS 2013, also available at http://arxiv.org/abs/1112.4419

[15] I. Giotis and V. Guruswami, Correlation clustering with a fixed number of clusters, Theory Comput. 2:249–266, 2006.

[16] J. Gramm, J. Guo, F. Hüffner, R. Niedermeier, Graph-modeled data clustering: Fixed parameter algorithms for clique generation, Theory Computing Syst. 38:373–392, 2005.

[17] J. Gramm, J. Guo, F. Hüffner, R. Niedermeier, Automated generation of search tree algorithms for hard graph modification problems, Algorithmica 39:321–347, 2004.

[18] J. Guo, A more effective linear kernelization for cluster editing, Theor. Comput. Sci. 410:718–726, 2009.

[19] F. Harary, On the notion of balance of a signed graph, Michigan Mathematical Journal 2(2):143–146, 1953.

[20] F. Hüffner, C. Komusiewicz, H. Moser, and R. Niedermeier, Fixed-parameter algorithms for cluster vertex deletion, Theory Computing Syst. 47(1):196–217, 2010.

[21] C. Komusiewicz and J. Uhlmann. Cluster editing with locally bounded modifications, Discret. Appl. Math., 160(15):2259–2270, 2012.

[22] R. Shamir, R. Sharan, and D. Tsur, Cluster graph modification problems, Discr. Appl. Math. 144(1–2):173–182, 2004.

[23] S. Wasserman and K. Faust, Social Network Analysis, Cambridge University Press, Cambridge, 1994.

[24] Wikipedia, http://en.wikipedia.org/wiki/Binomial_coefficient

Appendix

In Lemma 8, we show the time complexity of the search-tree algorithm is bounded by $O(\phi^{|U|+m})$. Here we show that the bound is tight for small k, i.e., for $k \in o(n^2)$ in Min-Sum 2-clustering (Corollary 10) and $k \in o(n^3)$ for Min-Square 2-clustering (Theorem 15).

As shown in the proof of Theorem 15, the number of leaf nodes of the search tree is $\sum_{i=0}^{m} \binom{|U|}{i}$. It is sufficient to show that

$$\max_m \binom{|U|}{m} \geq \phi^{|U|+m}. \tag{5}$$

Lemma 18: Suppose that β is a decreasing function of m. Let $\alpha = m/\beta$. The maximum of $\binom{\beta}{m}$ occurs at $m = m_0$, in which m_0 is the solution of

$$-\beta' \cdot \log(1-\alpha) - \log\frac{\alpha}{1-\alpha} = 0 \tag{6}$$

Proof: By String approximation,

$$\log \binom{\beta}{m} \approx \beta \cdot \lambda(\alpha), \tag{7}$$

in which $\lambda(\epsilon) \equiv -\epsilon \log(\epsilon) - (1-\epsilon)\log(1-\epsilon)$ is known as the binary entropy of ϵ [24]. In the following we use λ', β' and α' for the derivatives with respect to m.

$$\begin{aligned} \lambda' &= -(\alpha' \log \alpha + \alpha \cdot (\frac{\alpha'}{\alpha \ln 2})) - (-\alpha' \log(1-\alpha) + (1-\alpha) \cdot (\frac{-\alpha'}{(1-\alpha)\ln 2})) \\ &= -(\alpha' \log \alpha - \alpha' \log(1-\alpha) = -\alpha' \log \frac{\alpha}{1-\alpha}. \end{aligned}$$

Since $\alpha' = \frac{\beta - m\beta'}{\beta^2}$,

$$\begin{aligned} \beta \cdot \lambda' &= -(1 - \frac{m\beta'}{\beta}) \log \frac{\alpha}{1-\alpha} \\ &= -\log \frac{\alpha}{1-\alpha} + \alpha\beta' \cdot (\log \alpha - \log(1-\alpha)). \end{aligned}$$

Then,

$$\begin{aligned} & \frac{d}{dm} \log \binom{\beta}{m} \\ = \; & \beta' \cdot \lambda + \beta \cdot \lambda' \\ = \; & -\beta'(\alpha \log \alpha + (1-\alpha)\log(1-\alpha)) \\ & - \log \frac{\alpha}{1-\alpha} + \alpha\beta' \cdot (\log \alpha - \log(1-\alpha)) \\ = \; & -\beta' \log(1-\alpha) - \log \frac{\alpha}{1-\alpha} \end{aligned}$$

Since $\alpha = m/\beta$ is an increasing function with respect to m. By assumption $\beta' < 0$, the derivative is positive for sufficiently small α. When m goes from small to large, the derivative becomes negative when α is sufficiently large. Therefore, $\max_m \binom{\beta}{m}$ occurs at the value m_0 which makes the derivative zero, i.e., the solution of equation (6). □

Let $\beta = 2k/n - m$. First we note that $\max_m \binom{\frac{k}{(n+1)/2 - 2m} - m}{m} \geq \max_m \binom{\beta}{m}$, and we show (5) by a proof of $\binom{\beta(m_0)}{m_0} \geq \phi^{2k/n}$.

Apparently $\beta'(m) = -1$ and β is a decreasing function. By Lemma 18, the maximum occurs at m_0 which is the solution of (6). Let $\alpha_0 = \alpha(m_0)$ and $\beta_0 = \beta(m_0)$. Since $\beta' = -1$, we have

$$\log(1-\alpha_0) = \log \frac{\alpha_0}{1-\alpha_0}, \tag{8}$$

or equivalently $(1-\alpha_0)^2 = \alpha_0$. We have that $\alpha_0 = (3-\sqrt{5})/2$. By definition,

$$\lambda(\alpha_0) = -\alpha_0 \log \alpha_0 - (1-\alpha_0)\log(1-\alpha_0)$$

Then, since $\alpha = m/\beta$,

$$\begin{aligned} & \beta_0 \lambda(\alpha_0) \\ = \; & -\beta_0 \alpha_0 \log \alpha_0 - \beta_0 (1-\alpha_0) \log(1-\alpha_0) \\ = \; & -m_0 \log \alpha_0 - (\beta_0 - m_0) \log(1-\alpha_0) \\ = \; & -m_0 \log \frac{\alpha_0}{1-\alpha_0} - \beta_0 \log(1-\alpha_0) \\ = \; & -m_0 \log(1-\alpha_0) - \beta_0 \log(1-\alpha_0), \end{aligned}$$

in which the last step comes from (8). Since $-\log(1-\alpha_0) = \log \frac{1}{1-\alpha_0} = \log \frac{2}{\sqrt{5}-1} = \log \phi$, we obtain $\beta_0 \lambda(\alpha_0) = (m_0 + \beta_0) \log \phi$. By (7) and $\beta_0 + m_0 = 2k/n$,

$$\binom{\beta_0}{m_0} \approx 2^{\beta_0 \lambda(\alpha_0)} = (2^{\log \phi})^{2k/n} = \phi^{2k/n}.$$

□